TITLE:

国産数式処理システムGALにおけるユーザインターフェイスに関する一考察(数式処理と数学研究への応用)

AUTHOR(S):

高橋, 岳之; 松永, 稔雄; 長野, 英二

CITATION:

高橋, 岳之 ...[et al]. 国産数式処理システムGALにおけるユーザインターフェイスに関する一考察(数式処理と数学研究への応用). 数理解析研究所講究録 1991, 753: 144-152

ISSUE DATE:

1991-05

URL:

http://hdl.handle.net/2433/82078

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
第753巻 1991年 144-152

144

国産数式処理システム GAL における ユーザインターフェイスに関する一考察

高橋 岳之、 松永 稔雄、 長野 英二
(Takeyuki Takahashi) (Toshio Matsunaga) (Eiji Nagano)

豊橋技術科学大学

1 はじめに

現在、理化学研究所において GAL (General Algebraic Language/Laboratry) という数式処理システムが構築されている。我々はこの GAL のユーザインタフェース部分の開発を行なっており、その途中経過について報告する。

なお、現在行なっている作業は、GAL のワークステーションへの移植、入出力インタフェースの開発、及び積分求解の自動運用パッケージの開発などである。

2 KCL への移植

GAL は、メインフレーム上の CLISP (Cambridge LISP) で開発されている。しかし、最近のワークステーションに関する状況を考えると、GAL をワークステーション上に移植する必要性があると判断した。そこで、パブリックドメインソフトを使用するという前提のもと、KCL (Kyoto Common Lisp) へ GAL を移植することにした。当面は、KCL へ CLISP をインプリメントするという形態をとっている。現在7～8割程度が動作しているが、インタプリタで動作しているため非常に遅い。このため、パーサにコンパイル機能を付加し、かつ主な関数を直接 C で記述し処理速度の向上を行なっている。

3 入出力

いままでの数式処理システムは、一般に大型計算機の上で使用されていた。そのためユーザとのコミュニケーションはキャラクタベースが一般的であった。しかし、近年開発されたシステムにおいては、キャラクタベースの処理のみならず、ワークステーションの上での動作を考慮にいれたグラフィック機能などが取り入れられている。

我々は、これまでの数式処理システム上のユーザインタフェース (主に GAL) における不満と、これからのユーザインタフェースに必要と考えられる機能をあげ、その実現を試みる。

3.1 GAL におけるユーザインタフェース

現在の GAL におけるユーザインタフェースの状況を示す。

1. 数式の入力はすべて線形入力。
2. 入力式のチェックは数式処理システム内部で行ない、特殊なコマンドによるテキストの訂正、復帰。
3. 出力は線形出力。
4. コマンドにより入出力の切替を行ない、ファイルを操作する。
5. 計算した式を視覚的にとらえる機能 (グラフィック機能) が存在しない。
6. Mathematica, Maple などに見られるような、数式の TEX, C, Fortran などのテキストへの変換関数が存在しない。

3.2 これからの GAL におけるユーザインタフェース

では、GAL のためにユーザインタフェースの面から見て、どのような機能を持たせれば良いであろうか。将来的な機能としては、ライトペンのようなポインティングデバイスなどを用いて、紙と鉛筆を用いた感覚で数式の入力を行うといったようなことも考えている。しかし、ここでは現在あるワークステーション上において、今よりも使いやすいユーザインタフェースを開発することを考え、その改善箇所を以下に挙げる。

入力テキストの編集： 例えばごく一般の数式処理システムで用いられている線形入力において、数式の入力途中に自分のタイプミスなどを発見すると、その位置まで文字を削除しながらカーソルを移動し、さらに訂正した文字列を打たなければならない。このような作業は、入力文字列が短い場合はよいが入力文字列が長くなった場合非常に効率が悪い。そのため普通のエディタと同じようにカーソルの移動、任意の文字の削除、任意の位置への文字の挿入などができるようにする。

入力式文法チェック： 入力される式の括弧の対応などといった文法チェックは、数式処理システム内でおこなわれている。しかし、ユーザインタフェースシステム内でチェックを行なうことにより、数式処理システム自身の負荷を軽くする。これは、特に複数のユーザがそれぞれのユーザインタフェースシステムから、ひとつの数式処理システムプロセスを利用することができるようになった場合を想定している。

式の 2 次元表示： この機能は、現在のほとんどの数式処理システムでとり入れられているのと同様に数式処理システム内にスイッチを持たせ、2 次元出力のモードと線形出力のモードとの使いわけができるようにする。また、2 次元数式表示をした結果がひとつの項につき一行 (80 文字) で表示できないような場合は、線形表示をする。

ファイル読み込み・編集： GAL では、コマンドによってファイルを読み込みことが可能である。その機能に加え入力ウィンドウ内にテキストファイルを直接読み込む。

出力画面のファイルへの保存: 出力画面に表示された情報をすべてファイルに格納する機能を準備する。

入力履歴管理： 文字列として履歴を保存する機能を設ける。この履歴を用いて再入力も可能とするが、現在のところ環境については無視している。

他言語 (TEX, Fortran, C 言語など) のソースコード出力: 数式処理システムで計算をした結果をもとに、数値計算のプログラムを Fortran や、C 言語などで作成したり、あるいはその数式をもとに論文などの文章を書くときに必要となるテキスト変換機能を実現する。

数式の印刷形式表示： TEX を中間言語と仮定し、TEX のプレビュアなどを利用することにより、数式の印刷形式による表示を実現する。

2 次元/3 次元グラフ表示： この機能は、MACSYMA や *Mathematica* など現在の数式処理システムの多くで、すでに実現されている機能であるが、より細やかな制御を可能とし、複雑な動きなどもトレースできるようにする。

行列イメージ表示： これは、大規模な行列において、どのあたりに要素が集中しているかなど、行列の持つ性質を捉えやすくすることを目的として、行列をドットイメージにより表示できるようにする。

3.3 入出力インタフェースのシステム構成

現在作成中の入出力インタフェースの簡単な構成を図1に示す。

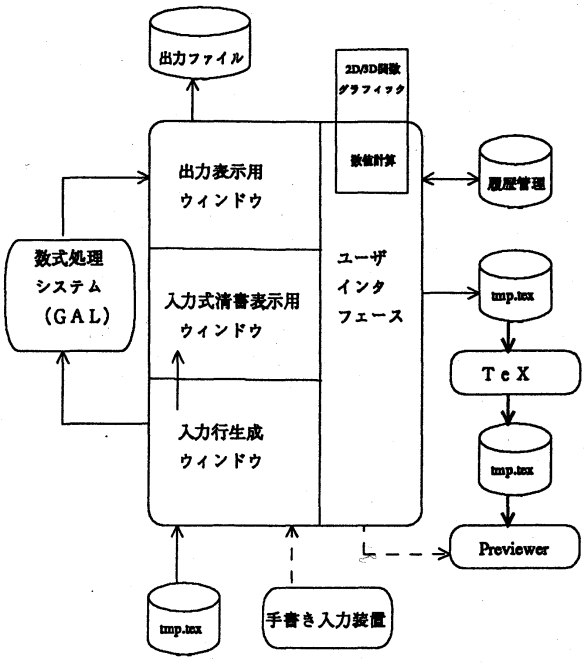

図 1: システム構成

入力行生成ウィンドウ： このウィンドウで基本的に式やコマンドの入力を行なう。このとき表示されている画面内では、基本的に Emacs と同等のキー操作により、カーソル移動や、文字の挿入、削除などができる。また、普通 X-Window 上で行なわれるテキストのカット&ペーストも同様に行なえる。

入力行確認のための清書印刷形式表示ウィンドウ： GAL に入力した式を転送する前にユーザインタフェース内で清書表示処理を行なう。すなわち、入力行生成ウィンドウ内で入力式を入力した後、<Ctrl>+L を入力することにより、ユーザインタフェース内でその式をキャラクタベースで清書表示する。

出力ウィンドウ： 出力ウィンドウには、一般のキャラクタベース端末でシステムを利用するときの画面そのものが表示される。つまり、システムからの出力 (入力のエコーバックも含む) が表示される。

カット&ペースト： 入力式の確認のための清書印刷形式表示ウィンドウ、システムからの出力を表示する出力ウィンドウの中にある部分式のような文字列もマウスによりカットバッファにコピーし、入力ウィンドウのカーソルの位置に挿入することができる。現在は表示された文字列をそのままの形式でしかコピーできないが、清書表示から線形表示への変換ルーチンを作成することにより、清書表示されている式を直接入力できるようにする。

ファイル操作機能： キーボードからの入力の代わりに指定したファイルから式やコマンドを入力することができる。また、出力ウィンドウ内のバッファに保持されているテキストをファイルに格納することもできる。

3.4 現状と課題

現在プロトタイプが、完成しているが、GAL の KCL への移植が完了していないため、動作確認などは完全には行なっていない。しかし、Maple をバックグラウンドにして処理を行なっているので、GAL の移植が完了次第、公開できるものと思われる。当面は、GAL の移植と並行して機能強化を進めていく。

4 積分データベースの自動運用

数式処理研究の初期には、発見的手法が多く使われていた。しかし、現在は解析的解法 (アルゴリズムによる解法) が研究の主流となっている。アルゴリズムの進歩は著しく、有理関数の不定積分や多項式の因数分解などの演算に対して解析的解法は有効である。しかしながら、現在もなお効率的なアルゴリズムが発見されてない演算も数多く存在している。例えば不定積分では、応用分野で扱われるような大規模なものや、結果が特殊関数になるような問題に対しては、現在のアルゴリズムでは一部を除いてほとんど解くことがで

きない。そこで、数式処理システムにも公式データベースを組み込むことにより、今まで解けなかった演算に対しても解を導き出せるようにすることを考える。

本研究では、図書館情報大学で作成された積分公式データベースを元に積分の求解パッケージを開発する。

4.1 システム構成

このパッケージの簡単な構成を図2に示す。

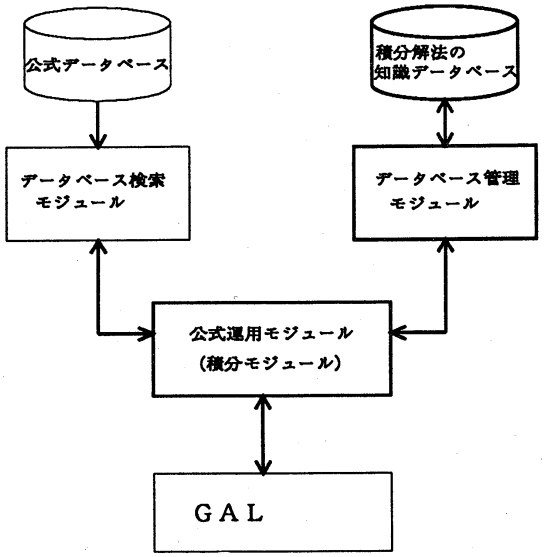

図 2: 積分公式運用モジュール

GAL 核部：数式処理システムの中心部で、主に因数分解や不定積分、GCD 演算などを行なうためのパッケージからなる。

積分公式データベース：積分公式が分野毎にまとめられ、区分化して格納されている。

公式データベース検索モジュール：検索要求に基づいて公式を検索するためのモジュール。

積分モジュール：積分公式と GAL の積分パッケージ、そして知識データベースを使い分けることにより積分を解く。

知識データベース：積分解法のノウハウを蓄積している。

知識データベース管理モジュール：被積分関数に対して、その解法を検索する。また積分を解くたびに、結果が出るまでの過程をデータベースに記録する。

4.2 公式の適用

現在、データベース検索により複数の公式が検索された場合には、どの公式を解として採用するかはユーザの判断に任せている。この作業を将来的には自動化する予定であるが、そのとき公式を選択する基準としては次のようなものが考えられる。

1. 結果の項数、次数を基準に選ぶ。
2. 任意の範囲での数値積分を求めてみる。
3. 知識データベース中の履歴を参照する。

4.3 ユーザインタフェース

公式が検索された場合、検索された公式と適用された結果が表示される。この画面上で、適用する公式を判断したり、あるいは残っている積分部分に対して改めて公式を適用する作業などを行なう。

4.4 現状と課題

現在開発しているシステムは、解析的に解けなかったものを発見的すなわち公式集を探すという手順を用いている。しかし、この方法だと解析的に解けないという定義が曖昧になるし、非常に時間が必要となる可能性がある。そこで、発見的なパッケージと解析的なパッケージをふたつに完全に分け、そして別のプロセスで動作することを考える。それにより、積分の求解のための平均時間を減らすことができると考える。そのためには、パターンマッチングの性能と、データベースの検索機能の強化を図る必要がある。また、履歴として保存されている知識データベースをどのように運用していくかの検討も必要であろう。